Optimizing Data Access With Visual Basic® 6.0 Part II

William R. Vaughn
Developer Trainer
Microsoft Technical Education


POWER UP
Windows DNA 2000
Readiness Conference
featuring SQL Server 2000

William R. Vaughn

Developer Trainer, Microsoft Corporation

Author:

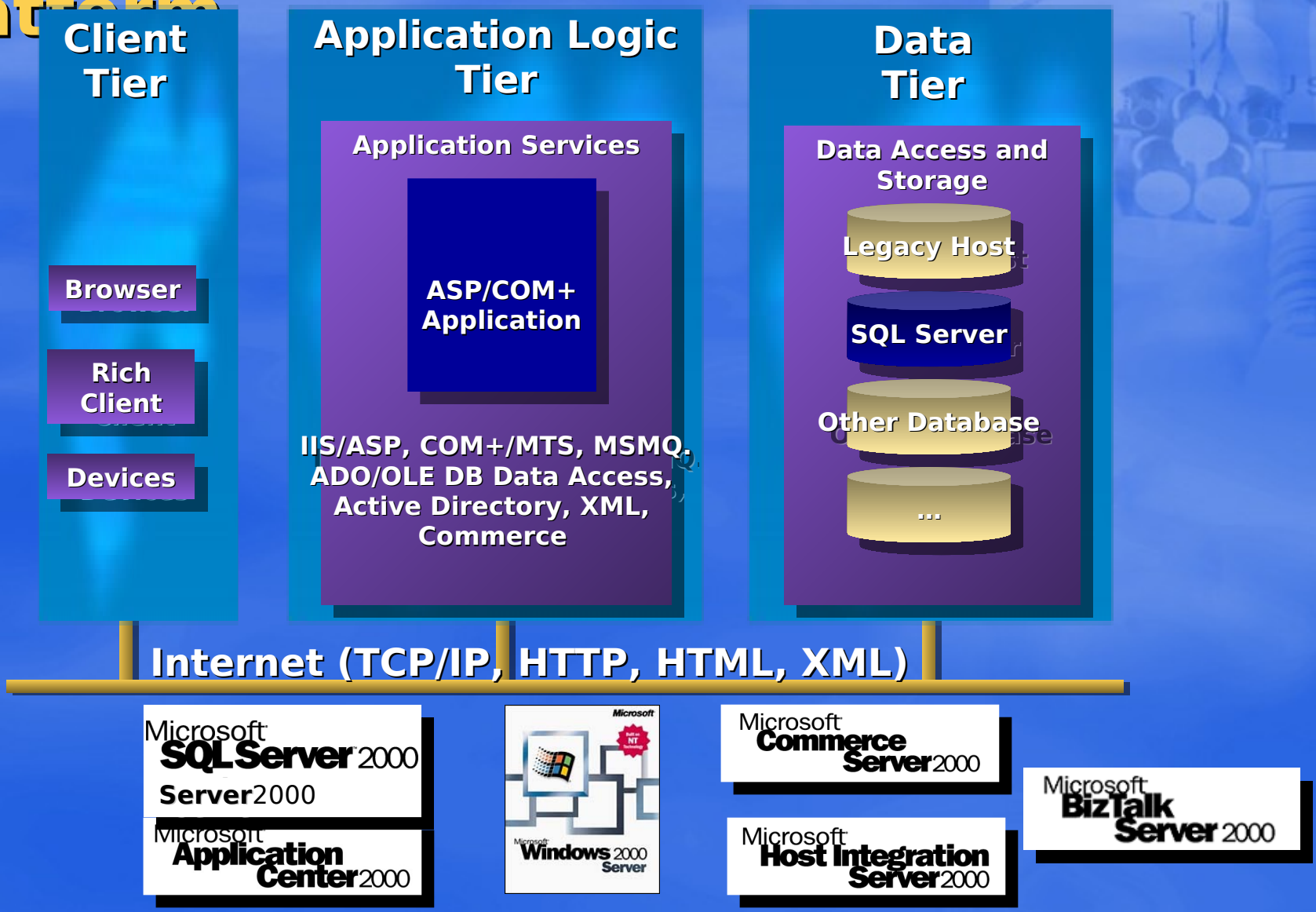
Hitchhiker's Guide to Visual Ba
SQL Server (now in 6 langua

billva@nwlink.com

www.betav.com

Windows DNA 2000

Next Generation Web Application Platform



Optimizing Data Access

Part II

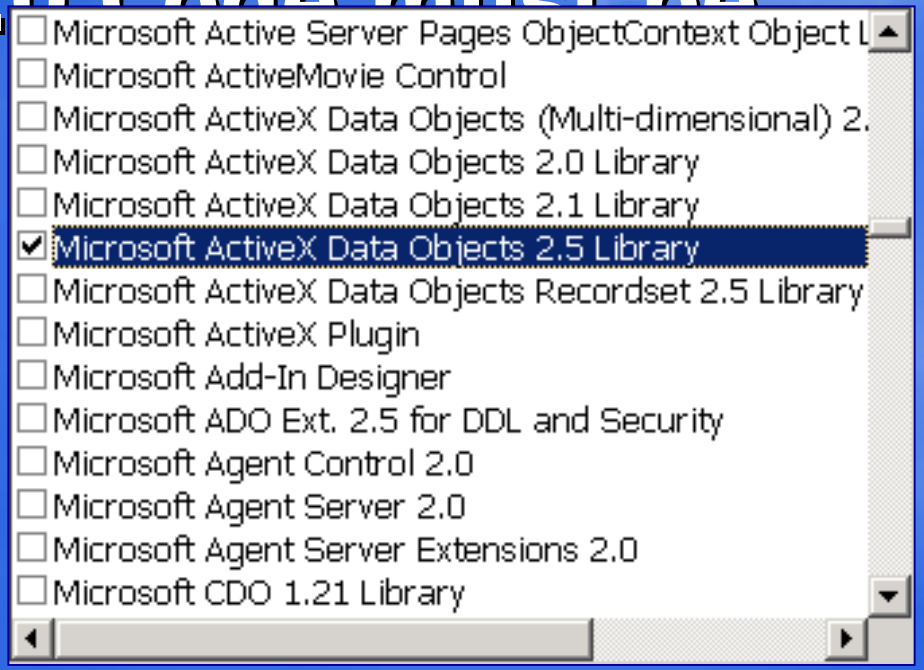
- Using ADO 2.5 with Visual Basic 6.0
- Passing data between tiers
- Writing smarter ADO code

Introduction To ADO 2.5

- ADO – a COM (object) interface to OLE DB
- Installed from Web: http://www.microsoft.com/data/download.htm

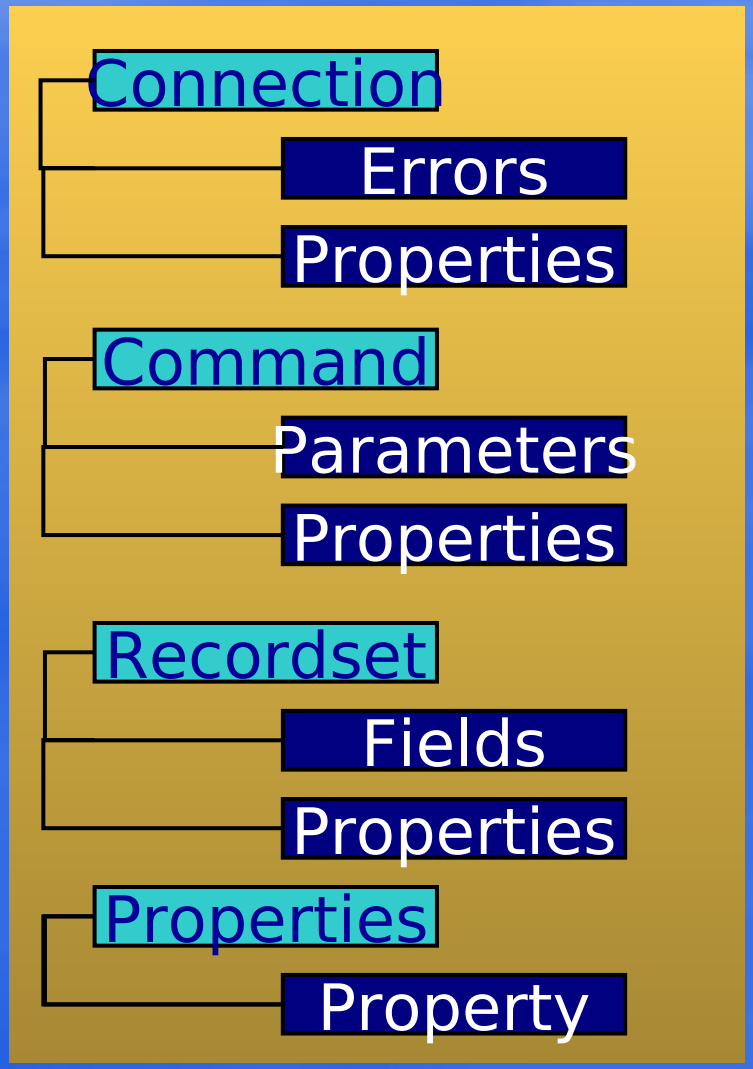
ADO 2.5 In Visual Basic 6.0

- Reference ADO 2.5 (MSADO15.DLL)
- Visual Basic 6.0 Code must be adapted
  - Data Environment Designer events
  - ADO Data control
  - Data Form Wizard
  - Visual Data Manager (source available)



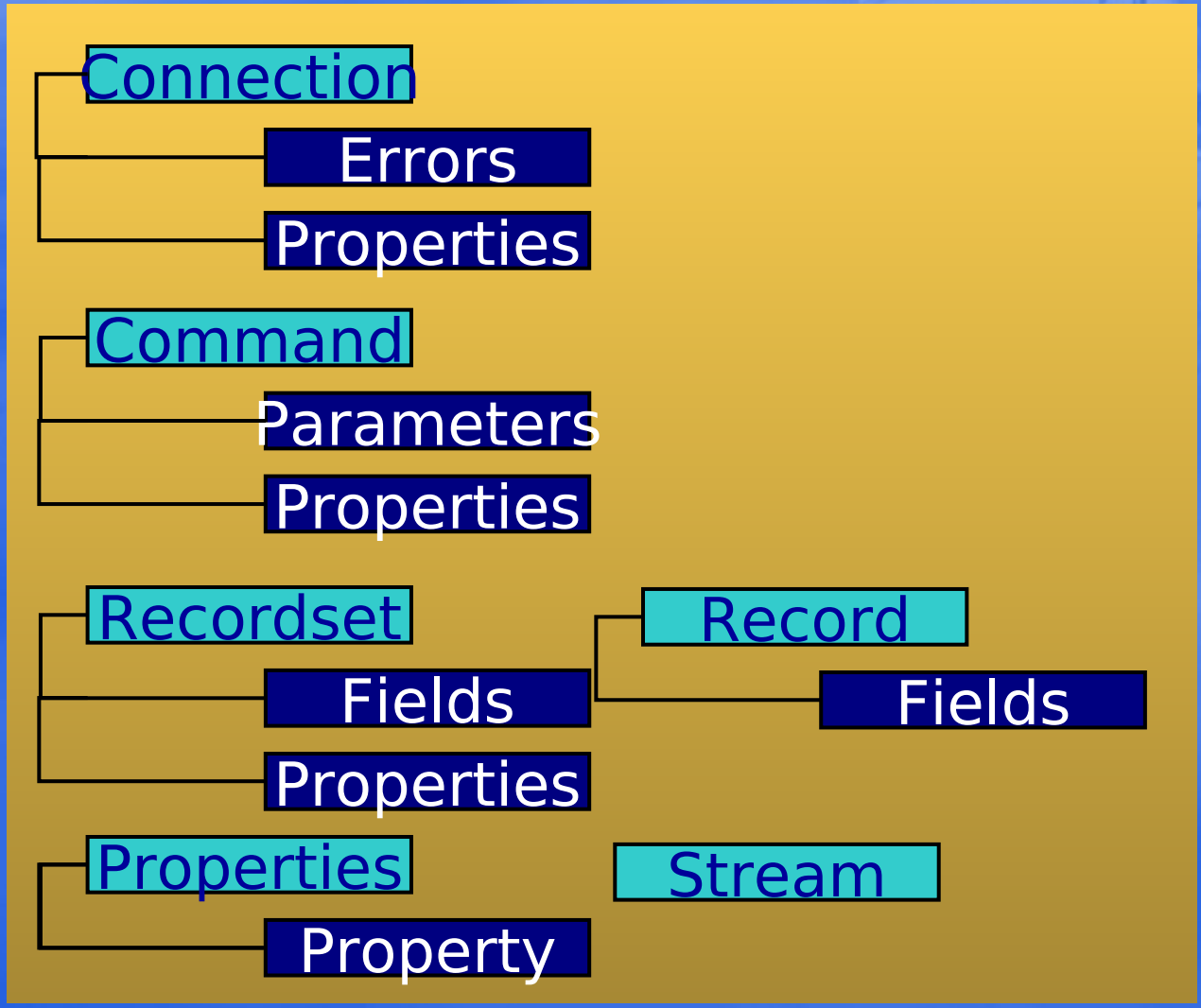
The ADO 2.0 Object Model

ADO 2.0



The ADO 2.5 Object Model

ADO 2.5



ADO 2.5 – New objects

- Record object - for directories, files, folders …
- Stream object - manages a binary byte stream
- To pass persisted Recordsets between layers
  - Recordset ➡ XML Stream XML Recordset
- URL Usage - alternative to named data store
  - Instead of connection strings, command text

ADO 2.5…

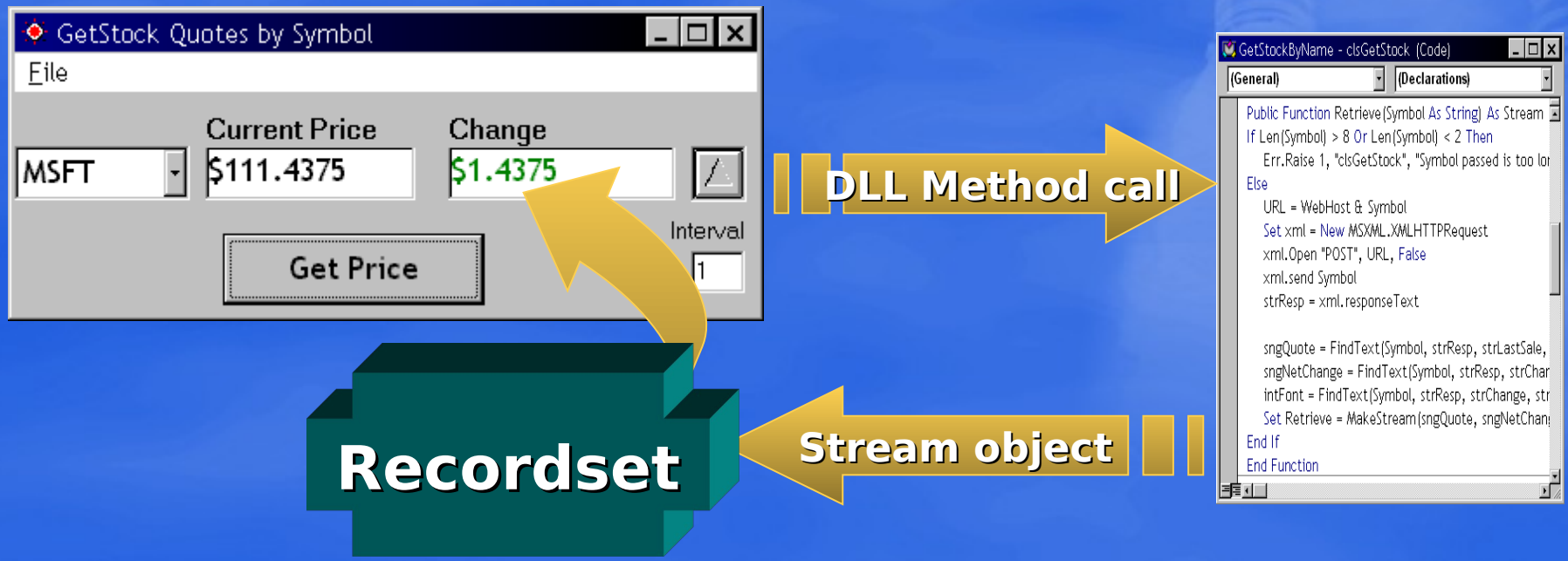
- New error handlers
  - Fewer “Error occurred” messages
  - Same number, different .Description
- XML persists hierarchical Recordsets
- Jet OLE DB provider for Jet 4.0 databases
  - Supports Table-Index paradigm

ADO 2.5 Implementation

- Passing data from tier-to-tier
  - Recordset objects
  - Streams
  - XML
  - Variant arrays
  - Delimited strings
  - User-defined data structures
  - PropertyBag objects

ADO 2.5 Implementation

- Passing Recordsets from tier-to-tier



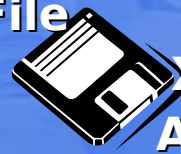
ADO Data Conversions

File

XML

ADTG

File

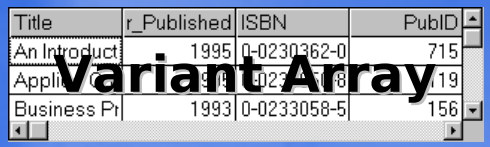


XML

ADTG

XML

Stream



Database

OUTPUT

Parms

Delimited String

PropertyBag

Offloading And Persisting

- GetRows ➨ variant array
- GetString delimited string
- Save method: persists to…
  - ADTG — ADO 1.0 on
  - XML— ADO 2.0
  - Hierarchical XML— ADO 2.5
  - Stream — ADO 2.5
- Update to data provider
- Execute method
  - INSERT, UPDATE SQL operations
  - Command object parameters

Extracting And Converting

- **Open method: accepts data from**
  - **Data source**
  - **ADTG - ADO 1.0 on**
  - **XML - ADO 2.1**
  - **Hierarchical XML - ADO 2.5**
  - **Stream - ADO 2.5**
  - **URL - ADO 2.5**
- **Command object**
  - **OUTPUT, return status parameters**

Passing ADO Data Between Tiers

- Consider memory expenses
  - Cache on server, client
  - Cursor location
- Consider LAN expenses
  - Unicode or ANSI
  - Padded structures or raw binary
- Consider session/server state
  - Loss of scalability
  - Slow user/system makes matters worse

Passing ADO Data Between Tiers

- Consider client app's intelligence
  - Ability to process 2.5 structures?
  - Code to bind, update, decipher, display structure
  - ADO capable?
- Consider developer's skill
- Limit size of rowset, locks, overhead

Recordset Services

- Manages (multiple) rowset/result set(s)
- Cursor services
  - Scrolling, binding, updatability
- Source/persistence from/to
  - Files, streams, Recordsets, providers
- Support by controls
  - Recordset, DataSource property
- Full data description language (DDL)

Passing Disjoint Recordsets

- Use Client(batch) CursorLocation
- Open Recordset as usual
- Assign (don't Set) ActiveConnection = Nothing
- Can be updatable, or RO
- Use RecordCount for row count (when available)
- Includes record status

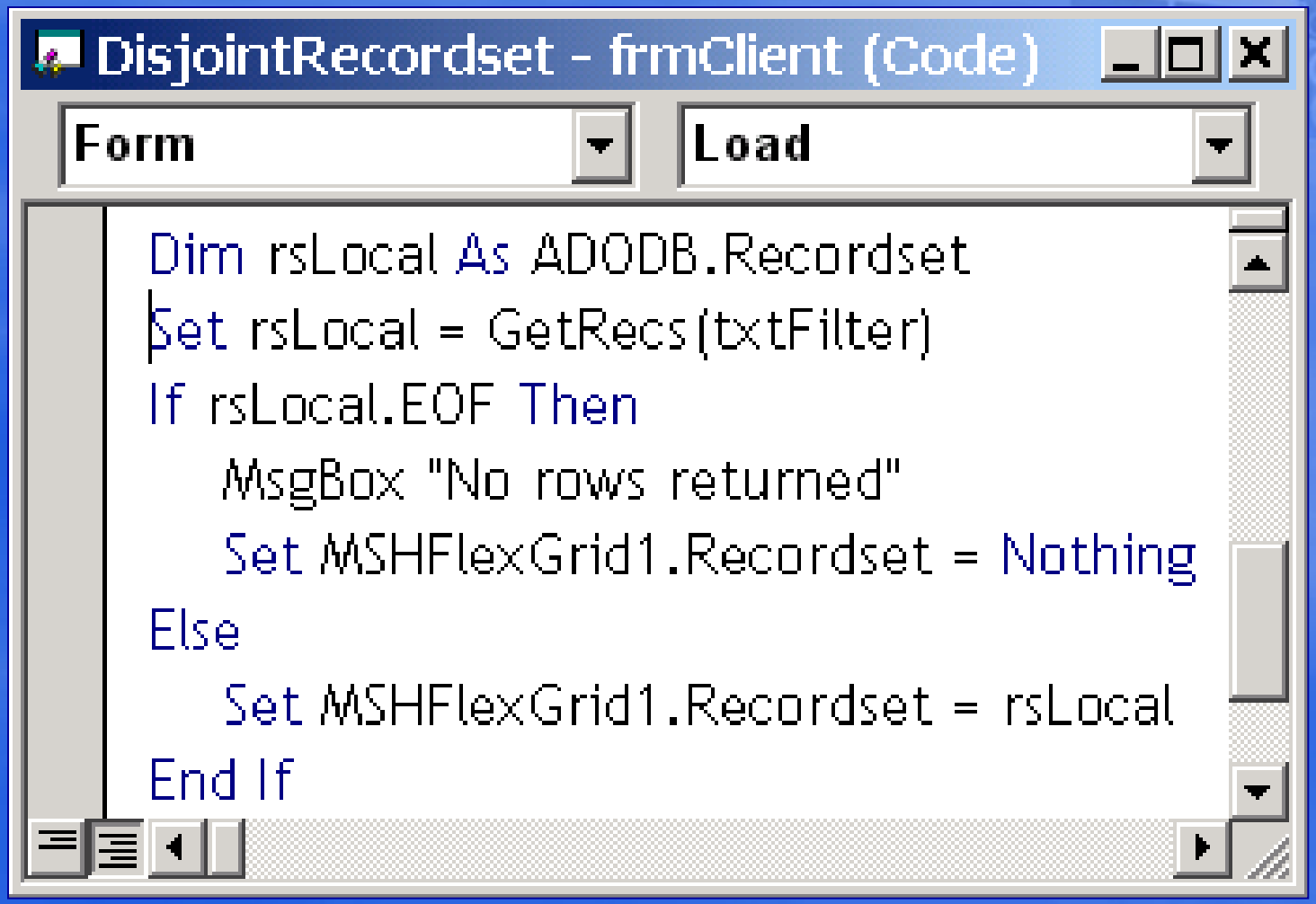
Passing Recordsets...
Server-side code

DisjointRecordset - GetRecsClass (Code)

(General) | GetRecs

```
Public Function GetRecs(Author As String) As Recordset
Dim rs As Recordset
OpenConnection
Set rs = New Recordset
With rs
    .CursorLocation = adUseClient
    .Open "select Au_id, Author, Year_Born " _
    & " from authors " _
    & " where author like '" & Author & "' ", cn, _
      Options:=adCmdText
    .ActiveConnection = Nothing
End With
```

Passing Recordsets...
Client-side code

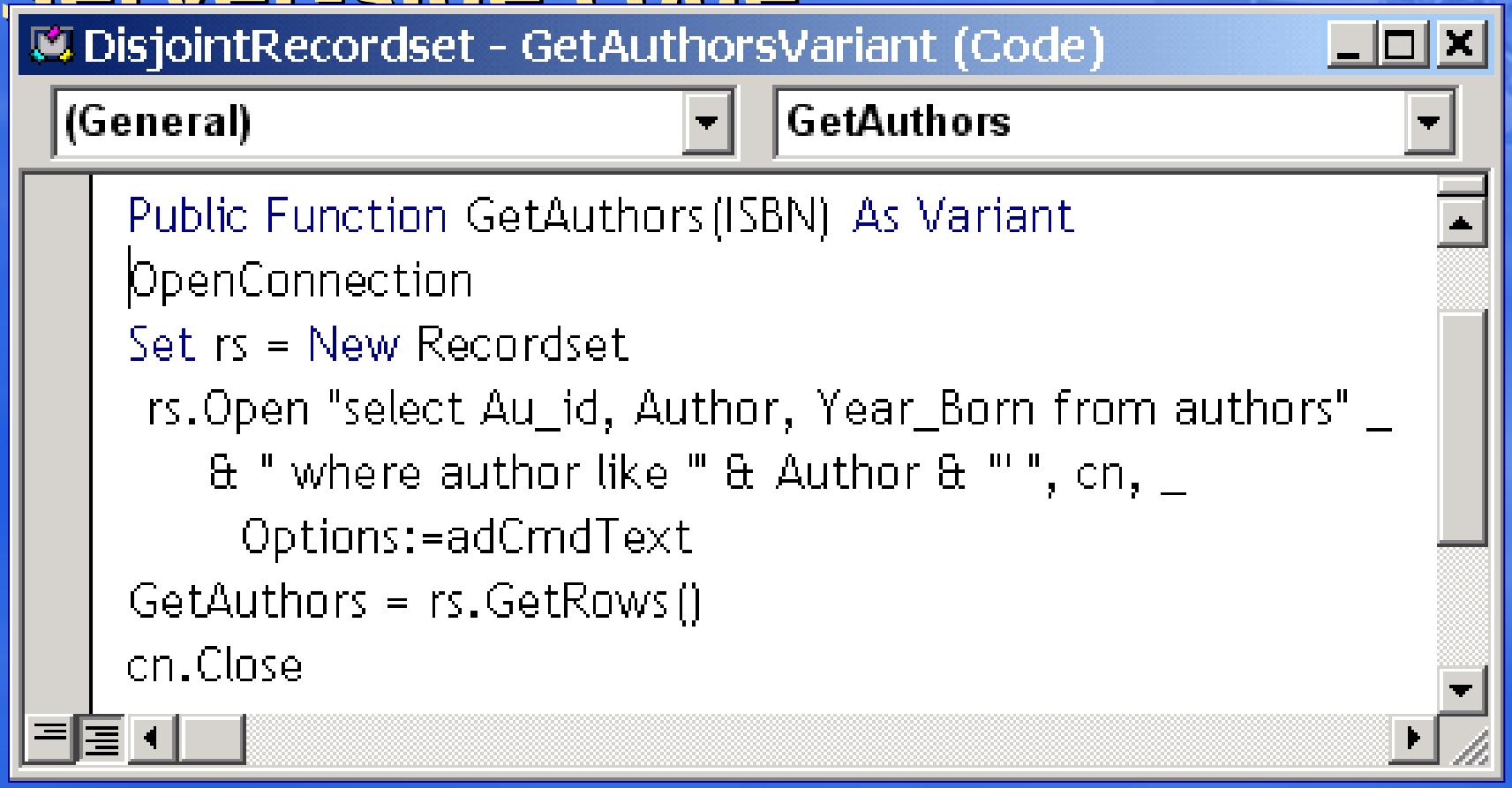


Passing Variant Arrays

- Use GetRows to offload data
- Returns variant array
  - Uses Unicode - 16-bits/character
  - High row overhead
    - Partial DDL
    - No Field Names
    - No Row status
  - Packed with filler-characters
    - Check NetMon
- Use UBound to determine record count

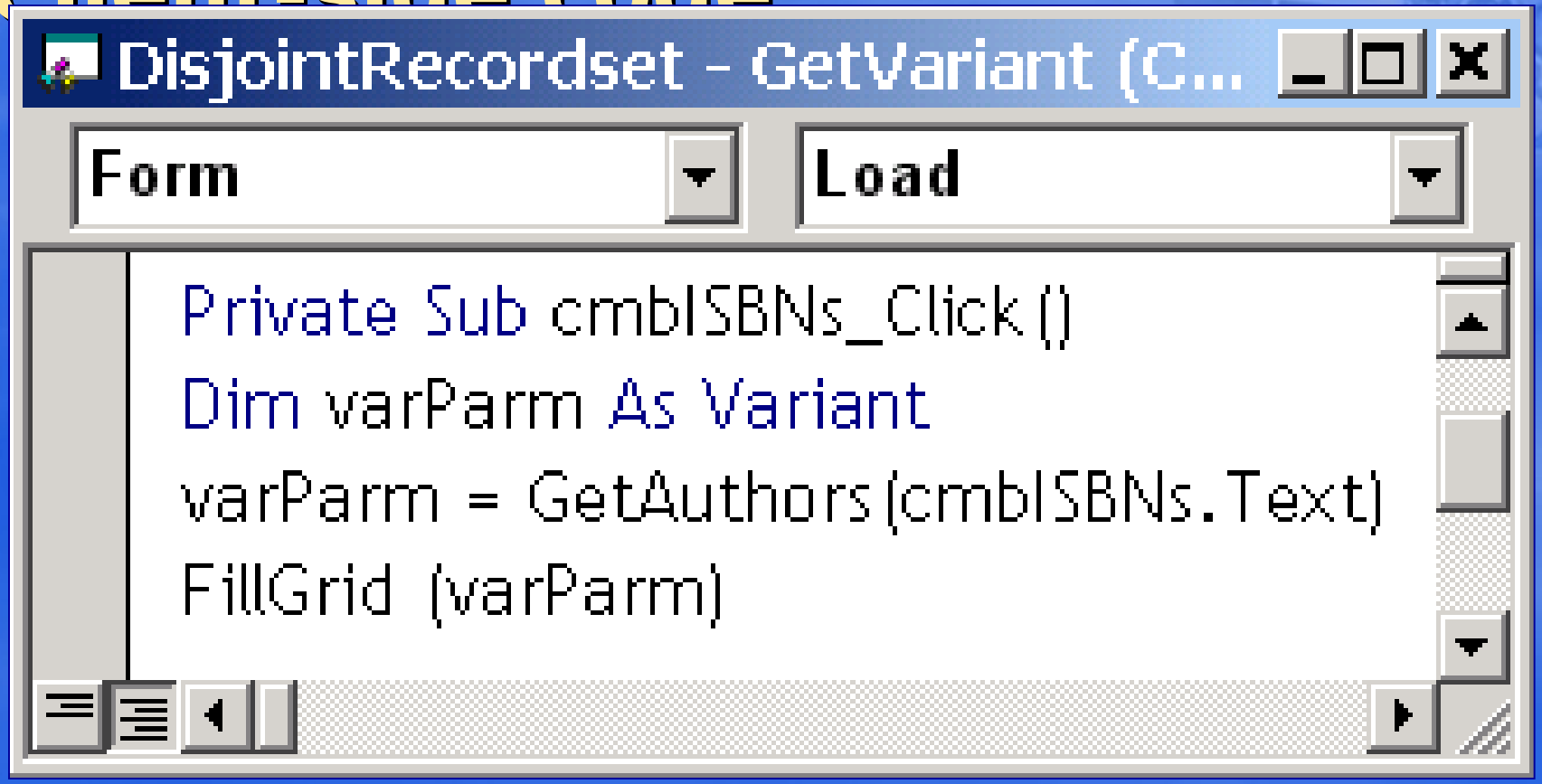
Passing Variant Arrays...

Server-side code



Passing Variant Arrays...

Client-side code



Passing Delimited String

- Use GetString to offload data
- Pseudo delimited strings

Tab Field delimiter

Using & Porting Gnu Cc, Version 2.3₮1993₮1-882114 1-9-1₮700₮50¬
Success With Visual C++₮1995₮1-8841330-9-6₮95₮44.95¬

CRLF Row delimiter

- Low row overhead - No DDL
- Assumes you know what's coming
- Can be applied to clip property
- Record count - on your own...

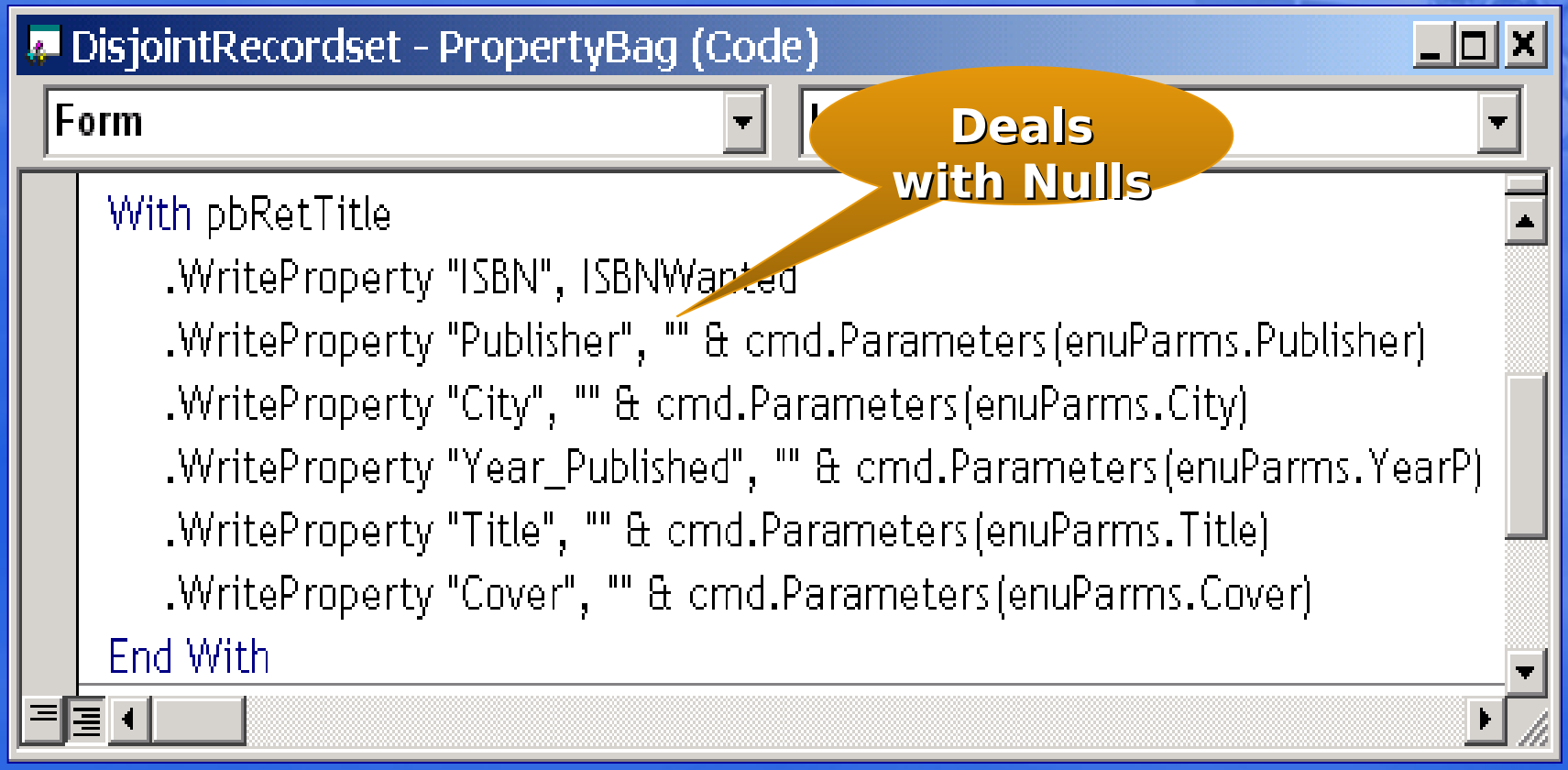
User-Defined
Data structures

- Pass
  - Single Type structure
  - Array of Type
- No ADO, COM, data control support
- Low row overhead - little DDL
- Record count - on your own…
- Assumes you know what's coming

Passing PropertyBag Objects

- Pass single Type structure or array of Type
- No ADO, control support
- COM structure referencing
- Low row overhead - little DDL
- Record count - on your own…
- Assumes you know what's coming

PropertyBag Objects
Server-side code



PropertyBag Objects
Client-side code

Optimizing For Performance

- Performance
  - What does it mean?
  - How is it measured?
- Achieving performance goals
  - Through smarter designs
  - Through smarter coding
  - Through smarter users

Optimal Performance

- Requires smarter
  - Designs
  - Implementations
  - Developers
  - Managers
  - World-class tools and engines

POWER UP

Microsoft